UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/911,663	07/24/2001	John Edward Ciolfi	04899-060001	3836

7590   01/09/2008

Thomas V. Smurzynski, Esq.
LAHIVE & COCKFIELD LLP
28 State Street
Boston, MA 02109-1784

EXAMINER
PILLAI, NAMITHA

ART UNIT	PAPER NUMBER
2173	

MAIL DATE	DELIVERY MODE
01/09/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

MAILED

JAN 0 9 2008

Technology Center 2100

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Application Number: 09/911,663
Filing Date: July 24, 2001
Appellant(s): CIOLFI, JOHN EDWARD

Kevin J. Canning
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 10/12/07 appealing from the Office action

mailed 9/8/06.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,937,257 B1	DUNLAVEY	8-2005
5,475,851	KODOSKY	12-1995

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form

the basis for the rejections under this section made in this Office action:

> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 33-44 and 46 are rejected under 35 U.S.C. 102(e) as being clearly

anticipated by U. S. Patent No. 6, 937, 257 B1 (Dunlavey).

As per claims 33 (method) and 46 (readable medium), Dunlavey discloses the

inventions substantially as claimed above. Dunlavey discloses the limitations of

receiving a plurality of user-defined block parameters (column 9, lines 55-67), teaching

that Dunlavey allows for multiple variables to be defined by the user. Dunlavey also

discloses processing the plurality of user-defined block parameters to produce a

plurality of run-time block parameters (column 3, lines 1-8), with the parameters defined

for the blocks by the user is optimized with its proper unit data to create an internal

representation of the user defined block parameter, creating the run-time block

parameter for modeling the graphical block diagram. All parameters that are defined by

the user for all components of the graphical block diagram have a representative run-

time optimized parameter that is created when the internal representation of the

graphical block diagram is generated. Dunlavey also teaches the grouping or pooling

together of like non-interfaced run-time block parameters to create a run-time parameter

expression for use during modeling, wherein Figure 4 lists the expression "SetDiscrete"

which includes a group of run-time block parameters that are grouped and share a

commonality of belonging to this group. Figure 4 is further taught to represent run-time

parameters and expressions as it is an internal representation (column 17, lines 63-65).

Dunlavey discloses multidimensional data types that represent various like variables,

which are then reused in relation to expressions and statements (column 3, lines 3-11).

As per claim 34, Dunlavey discloses mapping user defined block parameters into

an existing pool (column 7, lines 55-67), where user-defined parameters are received

and considered part of an existing pool to be evaluated on a periodic basis.

As per claim 35, Dunlavey discloses that the non-interfaced run-time block

parameters have stored values that differ from presented values (see col. 3, lines 1-7),

where a conversion process occurs from the presented value to the run-time

parameters.

As per claim 36, Dunlavey discloses that the non-interfaced run-time block

parameters are fixed point, where Figure 4 teaches "NamedConst" which is a

representation of a run-time block parameter that is fixed point.

As per claim 37, Dunlavey discloses translating at run-time constant parameter

values to an internal representation to enable increased pooling, where Figure 4

teaches a translated run-time parameter value that is an internal representation of "Unit"

which is used multiple times as needed for using units, thereby enabling increased

pooling (column 16, lines 38-45).

As per claim 38, Dunlavey discloses collecting constant portions of an expression containing an interfaced variable (column 24, lines 35-45), wherein discloses collecting the constant portions of an expression, as previous expressions that are calculated to a constant value and this constant value further used in an expression containing an interfaced variable, thereby teaching collecting the constant portions of an expression.

As per claim 39, Dunlavey discloses that the run-time block parameter is configured to return simulations results and automatically generated code that implements graphical block diagram model equations (column 19, lines 25-65).

As per claim 40, Dunlavey discloses that the code is automatically generated, the parameter expressions are maintained in the automatically generated code (column 19, lines 42-55).

As per claim 41, Dunlavey discloses that the parameter expressions contain interfaced variables that are updatable during modeling (column 25, lines 40-47).

As per claim 42, Dunlavey discloses converting to a relatively more compact representation portions of the parameter expressions that are constants while providing access to interface variables that are updatable (column 10, lines 45-50 and column 25, lines 40-50), where assignment expressions teaching assigning of a constant value to a variable and differential equation allows for manipulation of values with changes reflected in the graphs as the user updates the parameters.

As per claim 43, Dunlavey discloses that interfaced variables are updatable (column 25, lines 40-50).

As per claim 44, Dunlavey discloses that the updatable variables used in a

plurality of blocks are declared only once (column 19, lines 57-64), wherein teaching the

use of global variables which are variables that are declared once and is used in a

plurality of blocks or functions.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art
> are such that the subject matter as a whole would have been obvious at the time the invention was made to
> a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be
> negatived by the manner in which the invention was made.

Claims 16-24, 26-32 and 45 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Dunlavey and U. S. Patent No. 5, 475, 851 (Kodosky et al.), herein

referred to as Kodosky.

As per claims 16 (method) and 45 (readable medium), Dunlavey discloses a

method of mapping graphical block diagram block parameters in a graphical block

diagram modeling environment (see col. 9, lines 55-60), where these variables are

mapped into graphical block diagram for functions represented by the blocks of the

diagram. These parameters represent a value that is to be used during block diagram

execution, the parameters being used in the expressions that are executed as part of

the block diagram (column 9, lines 47-60). Dunlavey discloses receiving a user-defined

block parameter (see col. 9, lines 55-60) where the graphical block functions receive the

parameters that are defined by the user. Dunlavey discloses processing the user-

defined block parameter to produce a run-time block parameter for use during modeling

(column 3, lines 1-8), with the parameters defined for the blocks by the user is optimized

with its proper unit data to create an internal representation of the user defined block

parameter, creating the run-time block parameter for modeling the graphical block

diagram. Although Dunlavey does disclose grouping of parameters and the use of

global variables, Dunlavey does not clearly disclose that run-time block parameters

reduce memory requirements for executing a block diagram. Kodosky discloses using

global parameters in a graphical block diagram that is compiled and then executed,

thereby the parameters representing run-time block parameters (column 1, lines 28-32).

Kodosky further teaches that the use of global parameters reduces memory

requirements when executing the block diagram (column 59, lines 1-15), thereby

teaching a motivation for the user of global variables. It would have been obvious to

one skilled in the art, at the time of the invention to learn from Kodosky to use run-time

block parameters that reduce memory requirements for executing block diagrams. Both

Kodosky and Dunlavey disclose generating graphical block diagrams using parameters

that are to be executed. Kodosky further teaches that the use of global variables

reduces memory requirements for execution of the block diagram, with the storage and

execution process becoming more efficient (column 59, lines 1-15). Therefore, one

skilled in the art, at the time of the invention would have been motivated to learn from

Kodosky to use run-time block parameters that reduces memory requirements for

executing a block diagram.

As per claim 17, Dunlavey discloses a block method that inversely links or maps

the block run-time parameter to the user-defined block parameter (column 9, lines 7-

12), where the functions teach inverse linking of the run-time parameter to the user-

defined parameter.

As per claim 18, Dunlavey discloses the limitations of receiving a plurality of

user-defined block parameters (column 9, lines 55-67), teaching that Dunlavey allows

for multiple variables to be defined by the user.

As per claim 19, Dunlavey also discloses processing the plurality of user-defined

block parameter to produce a run-time block parameter (column 3, lines 1-8), with the

parameters defined for the blocks by the user is optimized with its proper unit data to

create an internal representation of the user defined block parameter, creating the run-

time block parameter for modeling the graphical block diagram.

As per claim 20, Dunlavey also discloses processing the plurality of user-defined

block parameter to optimally produce a single run-time block parameters (column 3,

lines 1-8), with the parameters defined for the blocks by the user is optimized with its

proper unit data to create an internal representation of the user defined block

parameter, creating the run-time block parameter for modeling the graphical block

diagram.

As per claim 21 and 27, Dunlavey discloses that the run-time block parameter is

configured to return simulations results and automatically generated code that

implements graphical block diagram model equations (column 19, lines 25-65).

As per claim 22, Dunlavey discloses mapping by discarding at least a portion of

the plurality of user-defined block parameters to reduce memory requirements (column

22, lines 45-55), where Dunlavey teaches only regarding distinct parameters within a

set of parameters, and thereby discarding at least a portion of the parameters, and

wherein the simulation requiring memory is reduced where the simulation only occurs

for the certain parameters that are not discarded.

As per claim 23, Dunlavey also teaches the grouping or pooling together of like

non-interfaced run-time block parameters to create a run-time parameter expression for

use during modeling, wherein Figure 4 lists the expression "SetDiscrete" which includes

a group of run-time block parameters that are grouped and share a commonality of

belonging to this group. Figure 4 is further taught to represent run-time parameters and

expressions as it is an internal representation (column 17, lines 63-65). The reference

to "SetDiscrete" allows for referencing a set of parameters, which would reduce

repetition of all variables contained within that category or set.

As per claim 24, Dunlavey discloses mapping user defined block parameters into

an existing pool (column 7, lines 55-67), where user-defined parameters are received

and considered part of an existing pool to be evaluated on a periodic basis.

As per claim 26, Dunlavey discloses mapping by translating the plurality of user-

defined block parameters based at least in part on type (column 11, lines 10-16).

As per claim 28, Dunlavey discloses that the code is automatically generated, the

parameter expressions are maintained in the automatically generated code (column 19,

lines 42-55).

As per claim 29, Dunlavey discloses that the parameter expressions contain

interfaced variables that are updatable during modeling (column 25, lines 40-47).

As per claim 30, Dunlavey discloses converting to a relatively more compact

representation portions of the parameter expressions that are constants while providing

access to interface variables that are updatable (column 10, lines 45-50 and column 25,

lines 40-50), where assignment expressions teaching assigning of a constant value to a

variable and differential equation allows for manipulation of values with changes

reflected in the graphs as the user updates the parameters.

As per claim 31, Dunlavey discloses that interfaced variables are updatable

(column 25, lines 40-50).

As per claim 32, Dunlavey discloses that the updatable variables used in a

plurality of blocks are declared only once (column 19, lines 57-64), wherein teaching the

use of global variables which are variables that are declared once and is used in a

plurality of blocks or functions.

(10) Response to Argument

1.   Dunlavey fails to disclose "pooling together like non-interfaced run-
        time block parameters"

Run time block parameters are the user defined block parameters that have been

processed or configured in way to provide an internal format through which the

parameters can be implemented. Any data provided by the user must be further

processed to be implemented into the system to configure and generate data

associated with the parameters. The user defined data will be converted to a run time

format before the data can be processed and executed within the system of Dunlavey.

Dunlavey discloses that as the user defined parameters are set by the user, the data set

are converted into an internal format (column 3, lines 1-23). Furthermore, Dunlavey discloses that as the model is constructed including the definition of the user parameters, the equations can be generated and the model itself can be tested using real-time emulation (column 4, lines 59-64). This testing of the model using real-time emulation discloses that the parameters are being executed as the parameters are defined by the user. The parameters take on an internal format and become run-time parameters so that the model generated can be tested. This suggests that execution is occurring as the user is defining the parameters. Furthermore, clearly the model along with its user defined parameter data must be executed and the parameters must be converted into a run time format in order to implement the model that is constructed. A run time block parameter is essentially an internal format of the parameter that is provided by the user. During execution, any data that must be executed is generated into an internal format from the user defined format in order to implement what the user has input. As is known in programming, code that is defined by the user is converted into an internal format during execution and to implement the code.

The multidimensional data type is one example of a variable that is a combination of pooled together like run time block parameters. The parameters provided in Figure 4 represent a variable used in code and include descriptions associated with the variables. The "SetDiscrete" variable includes multiple variables that are grouped together and are jointly distributed where multiple variables are represented in the one variable "SetDiscrete". There are other means for pooling together like parameters where a single common representation is used for multiple

parameters. For example, categorical distribution block and multivariate distribution

block both represent a single representation that is used to pool together multiple

variables across time. See column 8, lines 5-17. As per the Applicant's definition of

pooled together parameters (Appeal Brief, page 5, lines 1-3), a block is a single

common representation used for multiple parameters as is the case with the population

block. Dunlavey also discloses blocks that are used in the graphical model with the

blocks representing multiple variables that are associated with each other (column 6,

lines 40-50). These blocks are part of the graphical model and would be represented in

a run time format when the internal format is generated for the graphical model and also

when the generated internal mode is further compiled and executed. Furthermore,

Dunlavey discloses another representation where multiple variables are pooled

together. In this example, Dunlavey has clearly disclosed that the variables cannot be

modified by the user (column 10, lines 5-15). Dunlavey has provided multiple

representations where multiple variables that are associated with each other are pooled

together into one representation.

> 2. Dunlavey fails to disclose pooling "to reuse data for the like non-
> interfaced run-time block parameters together"

The parameters of Dunlavey including the pooled together non interfaced run

time block parameters can be reused. Dunlavey does disclose providing means for the

user to change values associated with variables where this involves reuse of variables

that are in the graphical model including pooled block variables which are

representations of multiple variables. Based on the user adjustments to variables the

same parameters can be used to calculate new results. See column 25, lines 40-51.

Furthermore there are representations of variables that are pooled together, the

variables being non interface variables. These include global variables which have

one instance but can be retrieved and reused multiple times within a procedure. The

purpose of the global variable is to provide one definition, with the variable being

accessible to multiple procedures and therefore reused multiple times. See column 10,

lines 5-15. The blocks of Dunlavey do represent multiple pooled parameters as shown

in Figure 3 where the block properties relate to multiple parameters that are associated

with each other. The population block clearly defines multiple variables that are

associated with each other and pooled together because of this relationship. See

column 6, lines 40-52.
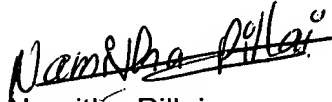
(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the

Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Namitha Pillai
Patent Examiner
Art Unit 2173
January 3, 2008

STEPHEN HONG
SUPERVISORY PATENT EXAMINER

Conferees:

Lynne H. Browne
Appeal Practice Specialist, TQAS
Technology Center 2100
January 3, 2008

Stephen Hong
Supervisory Patent Examiner
Art Unit 2178
January 3, 2008